# Towards Formalising Trace Equivalence
## for Global and Local Types

David Castro-Perez Francisco Ferreira Lorenzo Gheri
Nobuko Yoshida

Imperial College London

Verification of Session Types Workshop - 4th June 2020

# Certifying the Semantics of Communication

This work is part of a bigger project for certifying, and reasoning about, programs in distributed systems.

$$
\begin{tikzcd}
G \arrow[r, "\rho"] \arrow[d, "\upharpoonright"] & G^{c} \arrow[r, "\text{LTS}"] \arrow[d, "\upharpoonright^{c}"] & \text{Traces} \arrow[d, leftrightarrow, "="] \\
L \arrow[r, "\rho_L"] & L^{c} \arrow[r, "\text{LTS}"] & \text{Traces}
\end{tikzcd}
$$

- We formalize the meta-theory of multiparty session types[1].
- We use the Coq[2] Proof Assistant.

[1] Deniélou P.-M., Yoshida N. (2013) Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types. ICALP 2013.

[2] https://coq.inria.fr/

# Global and Local Types

Inductively defined by the following syntaxes:

$$
\begin{array}{rcll}
\mathsf{G} & ::= & \mathtt{end} & \text{end type} \\
& | & X & \text{variable} \\
& | & \mu X.\mathsf{G} & \text{recursion} \\
& | & \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i \in I} & \text{message}
\end{array}
$$

$$
\begin{array}{rcll}
\mathsf{L} & ::= & \mathtt{end} & \text{end type} \\
& | & X & \text{variable} \\
& | & \mu X.\mathsf{L} & \text{recursion} \\
& | & ![\mathsf{q}]; \{\ell_i(S_i).\mathsf{L}_i\}_{i \in I} & \text{send type} \\
& | & ?[\mathsf{p}]; \{\ell_i(S_i).\mathsf{L}_i\}_{i \in I} & \text{receive type}
\end{array}
$$

Types are assumed *closed* (no free variables) and recursion variables are always assumed *guarded* in types (namely types like $\mu X.X$ are not allowed).

# Projection

Projection Rules:

- $\mathtt{end}\upharpoonright \mathsf{r} = \mathtt{end}$; [PROJ-END]
- $X\upharpoonright \mathsf{r} = X$; [PROJ-VAR]
- $(\mu X.\mathsf{G})\upharpoonright \mathsf{r} = \mu X.(\mathsf{G}\upharpoonright \mathsf{r})$ [PROJ-REC]
- $\mathsf{r} = \mathsf{p}$ implies $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}\upharpoonright \mathsf{r} = ![\mathsf{q}]; \{\ell_i(S_i).\mathsf{G}_i\upharpoonright \mathsf{r}\}_{i\in I}$; [PROJ-SEND]
- $\mathsf{r} = \mathsf{q}$ implies $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}\upharpoonright \mathsf{r} = ?[\mathsf{p}]; \{\ell_i(S_i).\mathsf{G}_i\upharpoonright \mathsf{r}\}_{i\in I}$; [PROJ-RECV]
- $\mathsf{r} \neq \mathsf{p}$, $\mathsf{r} \neq \mathsf{q}$ and, for all $i, j \in I$, $\mathsf{G}_i\upharpoonright \mathsf{r} = \mathsf{G}_j\upharpoonright \mathsf{r}$; implies
  $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}\upharpoonright \mathsf{r} = \mathsf{G}_1\upharpoonright \mathsf{r}$; [PROJ-CONT]
- undefined otherwise.

# Warm-Up!

A global type for a simple protocol:

$$\mathsf{G} = \mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{q} \to \mathsf{r} : \ell'(S').\mathtt{end}$$

and its projection on participant q:

$$\mathsf{G}\upharpoonright\mathsf{q} = ?[\mathsf{p}];\ell(S).![\mathsf{r}];\ell'(S').\mathtt{end}$$

# Warm-Up!

A global type for a simple protocol:

$$\mathsf{G} = \mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{q} \to \mathsf{r} : \ell'(S').\mathtt{end}$$

and its projection on participant q:

$$\mathsf{G}\restriction\mathsf{q} = ?[\mathsf{p}];\ell(S).![\mathsf{r}];\ell'(S').\mathtt{end}$$

Do these global types have well defined projections?

$$\begin{aligned}\mathsf{p} \to \mathsf{q} :\{ \quad & \ell_1(\mathsf{oranges}).\mathtt{end}, \\ & \ell_2(\mathsf{bananas}).\mathsf{q} \to \mathsf{p} : \ell_3(\mathsf{pears}).\mathtt{end}\}\end{aligned}$$

$$\begin{aligned}\mathsf{p} \to \mathsf{q} :\{ \quad & \ell_1(\mathsf{oranges}).\mathtt{end}, \\ & \ell_2(\mathsf{bananas}).\mathsf{q} \to \mathsf{r} : \ell_3(\mathsf{pears}).\mathtt{end}\}\end{aligned}$$

$$\begin{aligned}\mathsf{p} \to \mathsf{q} :\{ \quad & \ell_1(\mathsf{oranges}).\mathsf{p} \to \mathsf{r} : \ell_3(\mathsf{pears}).\mathtt{end}, \\ & \ell_2(\mathsf{bananas}).\mathsf{q} \to \mathsf{r} : \ell_3(\mathsf{pears}).\mathtt{end}\}\end{aligned}$$

$$\begin{aligned}\mathsf{p} \to \mathsf{q} :\{ \quad & \ell_1(\mathsf{oranges}).\mathsf{q} \to \mathsf{r} : \ell_3(\mathsf{pears}).\mathtt{end}, \\ & \ell_2(\mathsf{bananas}).\mathsf{q} \to \mathsf{r} : \ell_3(\mathsf{pears}).\mathtt{end}\}\end{aligned}$$

# Warm-Up!

A global type for a simple protocol:

$$\mathsf{G} = \mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{q} \to \mathsf{r} : \ell'(S').\mathtt{end}$$

and its projection on participant q:

$$\mathsf{G}\upharpoonright\mathsf{q} = ?[\mathsf{p}];\ell(S).![\mathsf{r}];\ell'(S').\mathtt{end}$$

Do these global types have well defined projections?

$$\mathsf{p} \to \mathsf{q} : \{ \quad \ell_1(\text{oranges}).\mathtt{end}, \quad \ell_2(\text{bananas}).\mathsf{q} \to \mathsf{p} : \ell_3(\text{pears}).\mathtt{end}\}$$ **YES!**

$$\mathsf{p} \to \mathsf{q} : \{ \quad \ell_1(\text{oranges}).\mathtt{end}, \quad \ell_2(\text{bananas}).\mathsf{q} \to \mathsf{r} : \ell_3(\text{pears}).\mathtt{end}\}$$ **NO!**

$$\mathsf{p} \to \mathsf{q} : \{ \quad \ell_1(\text{oranges}).\mathsf{p} \to \mathsf{r} : \ell_3(\text{pears}).\mathtt{end}, \quad \ell_2(\text{bananas}).\mathsf{q} \to \mathsf{r} : \ell_3(\text{pears}).\mathtt{end}\}$$ **NO!**

$$\mathsf{p} \to \mathsf{q} : \{ \quad \ell_1(\text{oranges}).\mathsf{q} \to \mathsf{r} : \ell_3(\text{pears}).\mathtt{end}, \quad \ell_2(\text{bananas}).\mathsf{q} \to \mathsf{r} : \ell_3(\text{pears}).\mathtt{end}\}$$ **YES!**

# Certifying the Semantics of Communication

$$
\begin{tikzcd}
\mathsf{G} \arrow[r, "\rho"] \arrow[d, "\upharpoonright"] & \mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{LTS}"] \arrow[d, "\upharpoonright^{\mathsf{c}}"] & \text{Traces} \arrow[d, leftrightarrow, "="] \\
\mathsf{L} \arrow[r, "\rho_L"] & \mathsf{L}^{\mathsf{c}} \arrow[r, "\mathsf{LTS}"] & \text{Traces}
\end{tikzcd}
$$

- Multiparty Session Types ✓
- Coinductive Trees (equi-recursive point of view)
- Semantics by Traces

# An Equi-Recursive Viewpoint

"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.\mathsf{G}$ and $\mathsf{G}\{\mu X.\mathsf{G}/X\}$."

# An Equi-Recursive Viewpoint

"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.\mathsf{G}$ and $\mathsf{G}\{\mu X.\mathsf{G}/X\}$."

Example:
$\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X$ is "the same as" $\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,

# An Equi-Recursive Viewpoint

"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.\mathsf{G}$ and $\mathsf{G}\{\mu X.\mathsf{G}/X\}$."

Example:
$\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X$ is "the same as" $\mathsf{p} \to \mathsf{q} :\ell(S).(\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X)$,
which is "the same as" $\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :\ell(S).(\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X)$,

# An Equi-Recursive Viewpoint

"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.\mathsf{G}$ and $\mathsf{G}\{\mu X.\mathsf{G}/X\}$."

Example:
$\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X$ is "the same as" $\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,
which is "the same as" $\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,
which is "the same as"
$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,

# An Equi-Recursive Viewpoint

"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.\mathsf{G}$ and $\mathsf{G}\{\mu X.\mathsf{G}/X\}$."

Example:
$\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X$ is "the same as" $\mathsf{p} \to \mathsf{q} :\ell(S).(\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X)$,
which is "the same as" $\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :\ell(S).(\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X)$,
which is "the same as"
$\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :\ell(S).(\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X)$,
which is "the same as"
$\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{p} \to \mathsf{q} :(\mu X.\mathsf{p} \to \mathsf{q} :\ell(S).X)$,

# An Equi-Recursive Viewpoint

"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.\mathsf{G}$ and $\mathsf{G}\{\mu X.\mathsf{G}/X\}$."

Example:
$\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X$ is "the same as" $\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,
which is "the same as" $\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,
which is "the same as"
$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).(\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,
which is "the same as"
$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{p} \to \mathsf{q} : (\mu X.\mathsf{p} \to \mathsf{q} : \ell(S).X)$,
. . .

with a coinductive unrolling process!

# Coinductive Asynchronous Trees

Global Trees:

$$
\begin{array}{rcll}
\mathsf{G}^{\mathsf{c}} & ::= & \mathtt{end}^{\mathsf{c}} & \text{end type} \\
& \| & \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}^{\mathsf{c}}{}_i\}_{i \in I} & \text{message send} \\
& \| & \mathsf{p} \overset{\ell_{\bar{\imath}}}{\rightsquigarrow} \mathsf{q} : \{\ell_i(S_i).\mathsf{G}^{\mathsf{c}}{}_i\}_{i \in I} & \text{message receive}
\end{array}
$$

Coinductive unrolling $\_\rho\_$:

$$
\frac{}{\mathtt{end}\ \rho\ \mathtt{end}^{\mathsf{c}}} \qquad \frac{\mathsf{G}\{\mu X.\mathsf{G}/X\}\ \rho\ \mathsf{G}^{\mathsf{c}}}{\mu X.\mathsf{G}\ \rho\ \mathsf{G}^{\mathsf{c}}}
$$

$$
\frac{\forall i \in I. \mathsf{G}_i\ \rho\ \mathsf{G}^{\mathsf{c}}{}_i}{\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i \in I}\ \rho\ \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}^{\mathsf{c}}{}_i\}_{i \in I}}
$$

# Coinductive Asynchronous Trees

Global Trees:

$$\begin{array}{rcll} \mathsf{G}^{\mathsf{c}} & ::= & \mathtt{end}^{\mathsf{c}} & \text{end type} \\ & \| & \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}^{\mathsf{c}}{}_i\}_{i \in I} & \text{message send} \\ & \| & \mathsf{p} \overset{\ell_{\bar{\imath}}}{\rightsquigarrow} \mathsf{q} : \{\ell_i(S_i).\mathsf{G}^{\mathsf{c}}{}_i\}_{i \in I} & \text{message receive} \end{array}$$

Local Trees:

$$\begin{array}{rcll} \mathsf{L}^{\mathsf{c}} & ::= & \mathtt{end}^{\mathsf{c}} & \text{end type} \\ & \| & !^{\mathsf{c}}[\mathsf{p}]; \{\ell_i(S_i).\mathsf{L}^{\mathsf{c}}{}_i\}_{i \in I} & \text{send type} \\ & \| & ?^{\mathsf{c}}[\mathsf{q}]; \{\ell_i(S_i).\mathsf{L}^{\mathsf{c}}{}_i\}_{i \in I} & \text{receive type} \end{array}$$

# Certifying the Semantics of Communication

$$\begin{tikzcd}
\mathsf{G} \arrow[r, "\rho"] \arrow[d, "\upharpoonright"] & \mathsf{G^c} \arrow[r, "\mathsf{LTS}"] \arrow[d, "\upharpoonright^{c}"] & \text{Traces} \arrow[d, leftrightarrow, "="] \\
\mathsf{L} \arrow[r, "\rho_L"] & \mathsf{L^c} \arrow[r, "\mathsf{LTS}"] & \text{Traces}
\end{tikzcd}$$

- Multiparty Session Types ✓
- Coinductive Trees (equi-recursive point of view) ✓
- Semantics by Traces

# Asynchronous Semantics

p sends a message to q with label $\ell$:

$$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}}$$

We keep track of this communication with a queue:

$$\epsilon$$

# Asynchronous Semantics

p sends a message to q with label $\ell$:

$$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \xrightarrow{\mathsf{step}\ \ell} \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}}$$

We keep track of this communication with a queue:

$$\epsilon \xrightarrow{\mathsf{enqueue}} [(\ell, S)]$$

# Asynchronous Semantics

p sends a message to q with label $\ell$:

$$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \xrightarrow{\text{step } \ell} \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \xrightarrow{\text{step } \ell} \mathsf{G}^{\mathsf{c}}$$

We keep track of this communication with a queue:

$$\epsilon \xrightarrow{\text{enqueue}} [(\ell, S)] \xrightarrow{\text{dequeue}} \epsilon$$

# Asynchronous Semantics

p sends a message to q with label $\ell$:

$$\mathsf{p} \to \mathsf{q} :\ell(S).\mathsf{G}^{\mathsf{c}} \xrightarrow{\text{step } \ell} \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} :\ell(S).\mathsf{G}^{\mathsf{c}} \xrightarrow{\text{step } \ell} \mathsf{G}^{\mathsf{c}}$$

We keep track of this communication with a queue:

$$\epsilon \xrightarrow{\text{enqueue}} [(\ell, S)] \xrightarrow{\text{dequeue}} \epsilon$$

We have such a queue for each pair of participants.

# Queues for Asynchronous Local Coordination

p sends and q receive:

$$!^{\mathsf{c}}[\mathsf{q}];\ell(S).\mathsf{L}^{\mathsf{c}} \xrightarrow{\quad\text{step}\quad} \mathsf{L}^{\mathsf{c}}$$

$$Q(\mathsf{p},\mathsf{q}) = \epsilon \xrightarrow{\text{enqueue}} Q(\mathsf{p},\mathsf{q}) = [(\ell, S)] \xrightarrow{\text{dequeue}} Q(\mathsf{p},\mathsf{q}) = \epsilon$$

$$?^{\mathsf{c}}[\mathsf{p}];\ell(S).\mathsf{L}^{\mathsf{c}'} \xrightarrow{\quad\text{step}\quad} \mathsf{L}^{\mathsf{c}'}$$

**Queue Environments**

A *queue environment* is a finitely supported function $Q$ of type $\texttt{role} \times \texttt{role} \to \mathcal{W}$, where $\mathcal{W}$ is the set of finite words $w$ (*queues*) on the alphabet $\texttt{labels} \times \texttt{sorts}$.

# Global and Local Steps

p sends:

$$\begin{tikzcd}
\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{step}_1"] \arrow[d, "\upharpoonright_{\mathsf{p}}"] & \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{step}_2"] \arrow[dr, "\upharpoonright_{\mathsf{p}}"] & \mathsf{G}^{\mathsf{c}} \arrow[d, "\upharpoonright_{\mathsf{p}}"] \\
!^{\mathsf{c}}[\mathsf{q}];\ell(S).\mathsf{L}^{\mathsf{c}} \arrow[rr, "\mathsf{step}_1"] & & \mathsf{L}^{\mathsf{c}}
\end{tikzcd}$$

# Global and Local Steps

p sends:

$$
\begin{tikzcd}
\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{step}_1"] \arrow[d, "\upharpoonright_{\mathsf{p}}"] & \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{step}_2"] \arrow[dr, "\upharpoonright_{\mathsf{p}}"] & \mathsf{G}^{\mathsf{c}} \arrow[d, "\upharpoonright_{\mathsf{p}}"] \\
!^{\mathsf{c}}[\mathsf{q}];\ell(S).\mathsf{L}^{\mathsf{c}} \arrow[rr, "\mathsf{step}_1"] & & \mathsf{L}^{\mathsf{c}}
\end{tikzcd}
$$

q receives:

$$
\begin{tikzcd}
\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{step}_1"] \arrow[d, "\upharpoonright_{\mathsf{q}}"] & \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} : \ell(S).\mathsf{G}^{\mathsf{c}} \arrow[r, "\mathsf{step}_2"] \arrow[dl, "\upharpoonright_{\mathsf{q}}"] & \mathsf{G}^{\mathsf{c}} \arrow[d, "\upharpoonright_{\mathsf{q}}"] \\
?^{\mathsf{c}}[\mathsf{p}];\ell(S).\mathsf{L}^{\mathsf{c}\prime} \arrow[rr, "\mathsf{step}_2"] & & \mathsf{L}^{\mathsf{c}\prime}
\end{tikzcd}
$$

# “Local” Environments

We want to consider altogether the different local types involved in the communication.

**Environments for Local Types**

An *environment for local types* is a finitely supported function $E$ of type $\texttt{role} \to \texttt{l\_ty}^{c}$.

# "Local" Environments

We want to consider altogether the different local types involved in the communication.

**Environments for Local Types**

An *environment for local types* is a finitely supported function $E$ of type $\texttt{role} \to \texttt{l\_ty}^{c}$.

Why?!

# “Local” Environments

We want to consider altogether the different local types involved in the communication.

**Environments for Local Types**

An *environment for local types* is a finitely supported function $E$ of type $\texttt{role} \rightarrow \texttt{l\_ty}^{\mathsf{c}}$.

Why?!

It will be $E(\mathsf{p}) = \mathsf{L^c}_{\mathsf{p}}$, where $\mathsf{G^c} \restriction^{\mathsf{c}}_{\mathsf{p}} \mathsf{L^c}_{\mathsf{p}}$.

# One-Shot Projection

We get the one-shot projection of a global type both on environments of local types and on queue environments.

$$\begin{tikzcd} & \mathsf{G}^{\mathsf{c}} \arrow[dl, "\upharpoonright^{\mathsf{c}}"'] \arrow[dr, "\upharpoonright^{\mathsf{q}}"] & \\ E & & Q \end{tikzcd} \qquad \begin{tikzcd} \mathsf{G}^{\mathsf{c}} \arrow[d, "\upharpoonright\upharpoonright"] \\ (E, Q) \end{tikzcd}$$

# Step Results

**Theorem (Step Soundness)**

*If* $\mathsf{G}^{\mathsf{c}} \upharpoonleft\upharpoonright (E, Q)$ *and* $\mathsf{G}^{\mathsf{c}} \xrightarrow{step} \mathsf{G}^{\mathsf{c}\prime}$*, then there exist* $E'$ *and* $Q'$*, such that* $\mathsf{G}^{\mathsf{c}\prime} \upharpoonleft\upharpoonright (E', Q')$ *and* $(E, Q) \xrightarrow{step} (E', Q')$*.*

**Theorem (Step Completeness)**

*If* $\mathsf{G}^{\mathsf{c}} \upharpoonleft\upharpoonright (E, Q)$ *and* $(E, Q) \xrightarrow{step} (E', Q')$*, then there exists* $\mathsf{G}^{\mathsf{c}\prime}$*, such that* $\mathsf{G}^{\mathsf{c}\prime} \upharpoonleft\upharpoonright (E', Q')$ *and* $\mathsf{G}^{\mathsf{c}} \xrightarrow{step} \mathsf{G}^{\mathsf{c}\prime}$*.*

# Labelled Transition System

Keeping track of messages...

### Actions

An *action* $a$ is an object of the shape:

- either $\mathsf{pq}!(\ell, S)$ (send action),
- or $\mathsf{pq}?(\ell, S)$ (receive action).

### Traces

A *trace* $t$ is a coinductive, possibly infinite, stream of actions:

- $a_1 a_2 a_3 \ldots$ is a trace.

# Traces for Trees

p sends and q receives:

$$\mathsf{p} \to \mathsf{q} : \ell(S).\mathsf{G^c} \xrightarrow{\quad \mathsf{step}_1 \quad} \mathsf{p} \overset{\ell}{\rightsquigarrow} \mathsf{q} : \ell(S).\mathsf{G^c} \xrightarrow{\quad \mathsf{step}_2 \quad} \mathsf{G^c}$$

Let $t$ be a trace for $\mathsf{G^c}$,

$$\mathsf{pq}!(\ell, S)\mathsf{pq}?(\ell, S)t \xleftarrow[\quad \mathsf{step}_1 \quad]{} \mathsf{pq}?(\ell, S)t \xleftarrow[\quad \mathsf{step}_2 \quad]{} t$$

# Non-Determinism

Let us consider two different executions:

- $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G^c}_i\}_{i \in I} \xrightarrow{\text{step } \ell_1} \mathsf{p} \overset{\ell_1}{\rightsquigarrow} \mathsf{q} : \{\ell_i(S_i).\mathsf{G^c}_i\}_{i \in I} \xrightarrow{\text{step } \ell_1} \mathsf{G^c}_1$
- $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G^c}_i\}_{i \in I} \xrightarrow{\text{step } \ell_2} \mathsf{p} \overset{\ell_2}{\rightsquigarrow} \mathsf{q} : \{\ell_i(S_i).\mathsf{G^c}_i\}_{i \in I} \xrightarrow{\text{step } \ell_2} \mathsf{G^c}_2$

If $t_1$ is a trace admissible for $\mathsf{G^c}_1$ and $t_2$ is admissible for $\mathsf{G^c}_2$, both

- $\mathsf{pq}!(\ell_1, S_1)\mathsf{pq}?(\ell_1, S_1)t_1$ and
- $\mathsf{pq}!(\ell_2, S_2)\mathsf{pq}?(\ell_2, S_2)t_2$

are admissible for $\mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G^c}_i\}_{i \in I}$.

# Trace Equivalence

And finally...

# Trace Equivalence

And finally...

**Theorem (Trace Equivalence)**

*If $\mathsf{G}^{\mathsf{c}} \upharpoonright^{\mathsf{c}} E$ then the set of traces admissible for $\mathsf{G}^{\mathsf{c}}$ is equal to the set of traces admissible for $E$.*

# Certifying the Semantics of Communication

$$\begin{tikzcd} \mathsf{G} \arrow[r, "\rho"] \arrow[d, "\upharpoonright"] & \mathsf{G^c} \arrow[r, "\mathsf{LTS}"] \arrow[d, "\upharpoonright^c"] & \mathsf{Traces} \arrow[d, leftrightarrow, "="] \\ \mathsf{L} \arrow[r, "\rho_L"] & \mathsf{L^c} \arrow[r, "\mathsf{LTS}"] & \mathsf{Traces} \end{tikzcd}$$

- Multiparty Session Types ✓
- Coinductive Trees (equi-recursive point of view) ✓
- Semantics by Traces ✓(almost)

# Things We Used

Formalisation in the Coq[3] Proof Assistant, in particular we have used:

- the SSReflect[4] proof language;
- the Mathematical Components[5] libraries;
- the PaCo library for parametrized coinduction[6].

[3] https://coq.inria.fr/

[4] https://coq.inria.fr/refman/proof-engine/ssreflect-proof-language.html

[5] https://math-comp.github.io/

[6] https://github.com/snu-sf/paco

# Conclusion

**Things we have got:**

- a formalisation of the metatheory of mutliparty session types in Coq
- two birds with a (coinductive) stone: equi-recursion and no bindings
- (non-deterministic) semantics through labelled transition systems
- types that are ready for typing!

**From here what?**

- a more comprehensive version of MPST (e.g., with merge operator)
- communicating finite state automata
- ... the future is unwritten!

# Conclusion

**Things we have got:**

- a formalisation of the metatheory of mutliparty session types in Coq
- two birds with a (coinductive) stone: equi-recursion and no bindings
- (non-deterministic) semantics through labelled transition systems
- types that are ready for typing! Please attend Francisco's talk! :)

**From here what?**

- a more comprehensive version of MPST (e.g., with merge operator)
- communicating finite state automata
- ... the future is unwritten!

Thank You!